Information Bottleneck and Deep Learning

Haoran Sun

Information bottleneck

- A network rids noisy input data of extraneous details as if by squeezing the information through a bottleneck, retaining only the features most relevant to general concepts.

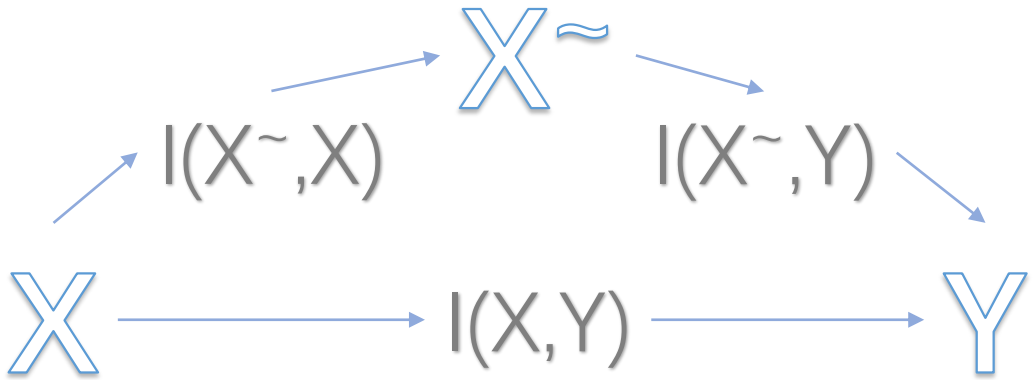
The Information Bottleneck Method

Naftali Tishby, Fernando C. Pereira, and William Bialek

Relevant information

- We define the relevant information in a signal $x \in X$ as being the information that this signal provides about another signal $y \in Y$. Understanding the signal x requires more than just predicting y, it also requires specifying which features of X play a role in the prediction.
- We formalize this problem as that of finding a short code for X that preserves the maximum information about Y .
- That is, we squeeze the information that X provides about Y through a 'bottleneck' formed by a limited set of codewords $\tilde{X}$. This approach yields an exact set of self consistent equations for the coding rules $X \rightarrow \tilde{X}$ and $\tilde{X} \rightarrow Y$.

- The relevance variable, denoted here by Y , must not be independent from the original signal X, namely they have positive mutual information I(X; Y).



Information bottleneck principle

- The relevance variable, denoted here by Y , must not be independent from the original signal X, namely they have positive mutual information I(X; Y).

$$I(X;Y) = D_{KL}[p(x,y)||p(x)p(y)] = \sum_{x\in X, y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$
$$= \sum_{x\in X, y\in Y} p(x,y)\log\left(\frac{p(x|y)}{p(x)}\right) = H(X) - H(X|Y)\,,$$

- We would like our relevant quantization X˜ to compress X as much as possible. In contrast to the rate distortion problem, however, we now want this quantization to capture as much of the information about Y as possible.

- The amount of information about Y in X˜ is given by

$$I(\tilde{X};Y) = \sum_y \sum_{\tilde{x}} p(y,\tilde{x}) \log \frac{p(y,\tilde{x})}{p(y)p(\tilde{x})} \leq I(X;Y).$$

- We can find the optimal assignment by minimizing the functional

$$\mathcal{L}[p(\tilde{x}|x)] = I(\tilde{X};X) - \beta I(\tilde{X};Y)$$

Deep Learning and the Information Bottleneck Principle

Naftali Tishby, Noga Zaslavsky

DL and IB

- The goal of any supervised learning is to capture and efficiently represent the relevant information in the input variable about the output-label-variable. Namely, to extract an approximate minimal sufficient statistics of the input with respect to the output.
- The information theoretic interpretation of minimal sufficient statistics suggests a principled way of doing that: find a maximally compressed mapping of the input variable that preserves as much as possible the information on the output variable.
- This is precisely the goal of the Information Bottleneck (IB) method.

- Basic questions about the design principles of deep networks, the optimal architecture, the number of required layers, the sample complexity, and the best optimization algorithms, are not well understood.

Relevant information in DNNs

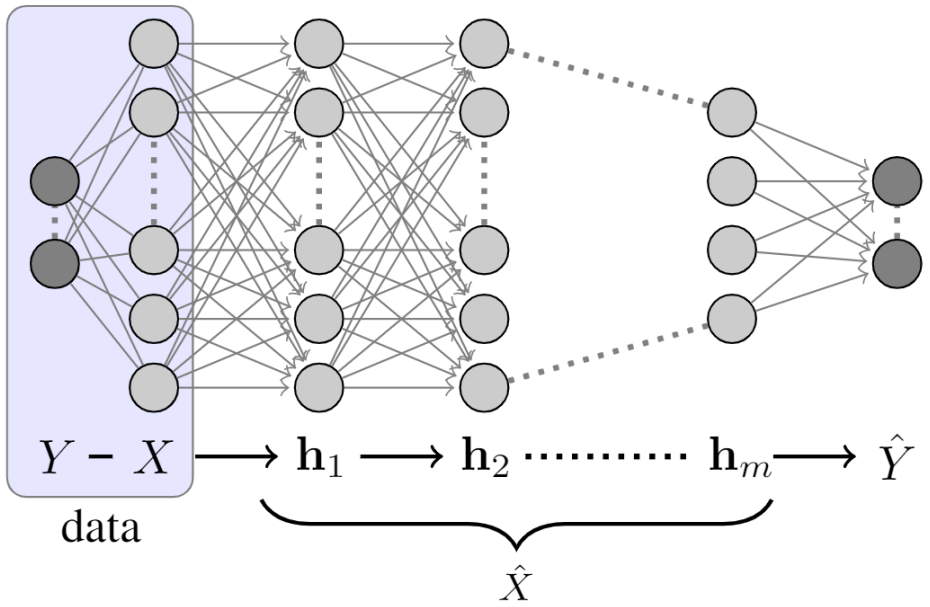


Fig. 1. An example of a feedforward DNN with m hidden layers, an input layer X and an output layer $\hat{Y}$. The desired output, Y, is observed only during the training phase through a finite sample of the joint distribution, $p(X, Y)$, and is used for learning the connectivity matrices between consecutive layers. After training, the network receives an input X, and successively processes it through the layers, which form a Markov chain, to the predicted output $\hat{Y}$. $I(Y;\hat{Y})/I(X;Y)$ quantifies how much of the relevant information is captured by the network.

Information characteristics of the layers

- Each layer in a DNN processes inputs only from the previous layer, which means that the network layers form a Markov chain.
- An immediate consequence of the DPI is that information about Y that is lost in one layer cannot be recovered in higher layers. Namely, for any i⩾j it holds that

$$I(Y;X) \geq I(Y;\mathbf{h}_j) \geq I(Y;\mathbf{h}_i) \geq I\left(Y;\hat{Y}\right)$$

- Achieving equality here is possible if and only if each layer is a sufficient statistic of its input. By requiring not only the most relevant representation at each layer, but also the most concise representation of the input, each layer should attempt to maximize $I(Y;\mathbf{h}_i)$ while minimizing $I(\mathbf{h}_{i-1};\mathbf{h}_i)$ as much as possible.

Finite samples and generalization bounds

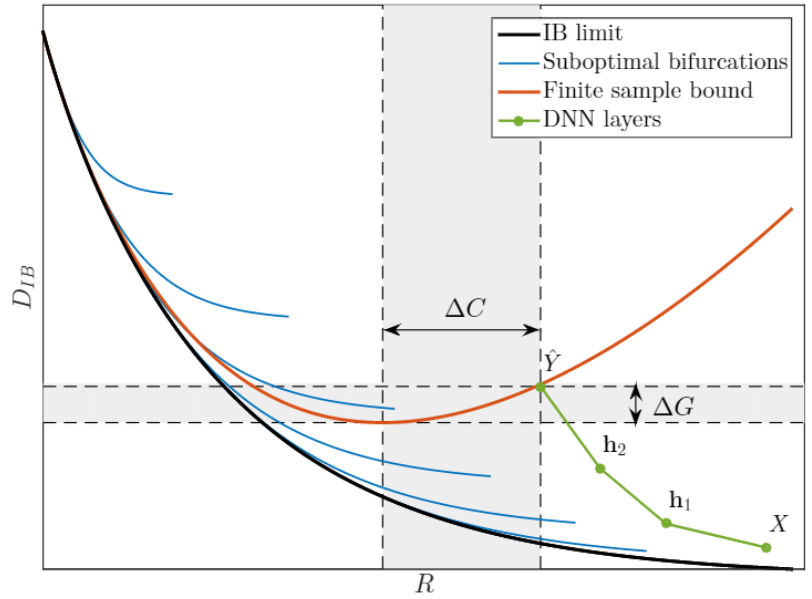


Fig. 2. A qualitative information plane, with a hypothesized path of the layers in a typical DNN (green line) on the training data. The black line is the optimal achievable IB limit, and the blue lines are sub-optimal IB bifurcations, obtained by forcing the cardinality of $\hat{X}$ or remaining in the same representation. The red line corresponds to the upper bound on the *out-of-sample* IB distortion (mutual information on Y), when training from a finite sample. While the training distortion may be very low (the green points) the actual distortion can be as high as the red bound. This is the reason why one would like to shift the green DNN layers closer to the optimal curve to obtain lower complexity and better generalization. Another interesting consequence is that getting closer to the optimal limit requires stochastic mapping between the layers.

A method for evaluating the network. Let N be a given DNN, and denote by D_N the IB distortion of the network's output layer, i.e. $I(X;Y|\hat{Y})$,and by R_N the representational complexity of the output layer, i.e.$I(X;\hat{Y})$. We can now define two measures for the performance of the network in terms of prediction and compression. The first one is the generalization gap,

$$\Delta G = D_N - D_{IB}^*(n)$$

which bounds the amount of information about Y that the network did not capture although it could have. The second measure is the complexity gap,

$$\Delta C = R_N - R^*(n)$$

which bounds the amount of unnecessary complexity in the network.

Finite samples and generalization bounds

- The empirical input layer of a DNN alone cannot guarantee good generalization even though it contains more information about the target variable Y than the hidden layers, as its representation of the data is too complex. Compression is thus necessary for generalization.

- Here is no reason to believe that current training algorithms for DNNs will reach the optimal point of the IB finite sample bound.

- However, we do believe that the improved feature detection along the network's layers corresponds to improvement on the information plane in this direction. In other words, when placing the layers of a trained DNN on the information plane, they should form a path similar to the green curve in figure 2.

- It is thus desirable to find new training algorithms that are based on the IB optimality conditions and can shift the DNN layers closer to the optimal limit.

Opening the Black Box of Deep Neural Networks via Information

Ravid Schwartz-Ziv, Naftali Tishby

Main results

- (i) the Stochastic Gradient Decent (SGD) optimization has two main phases. In the first and shorter phase the layers increase the information on the labels(fitting), while in the second and much longer phase the layer reduce the information on the input(compression phase).
- (ii) The converged layers lie on or very close to the IB theoretical bound.
- (iii) The main advantage of the hidden layers is computational, as they dramatically reduce the stochastic relaxation times.
- (iv) The hidden layers appear to lie close to critical points on the IB bound.

Snapshots of layers in the information plane

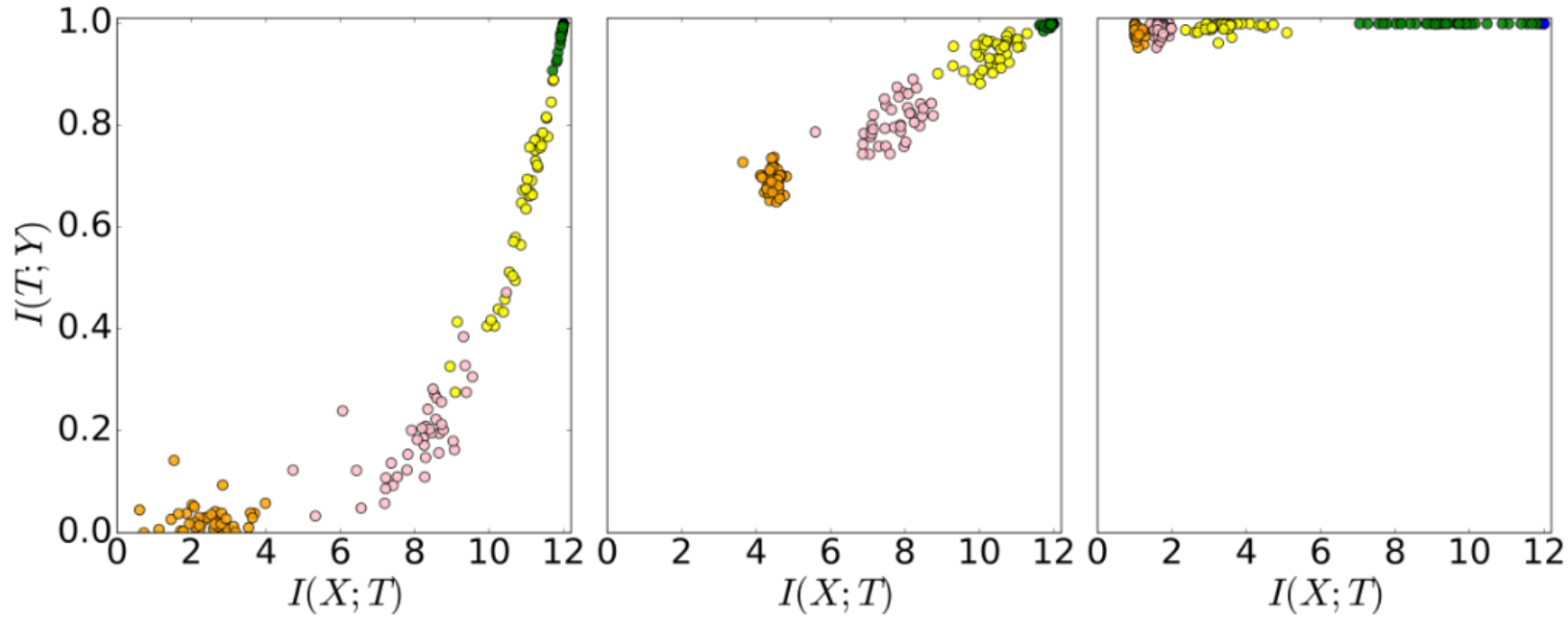


Figure 2: Snapshots of layers (different colors) of 50 randomized networks during the SGD optimization process in the *information plane* (in bits): **left** - with the initial weights; **center** - at 400 epochs; **right** - after 9000 epochs. The reader is encouraged to view the full videos of this optimization process in the *information plane* at *https://goo.gl/rygyIT* and *https://goo.gl/DQWuDD*.

The evolution of the layers in the info plane

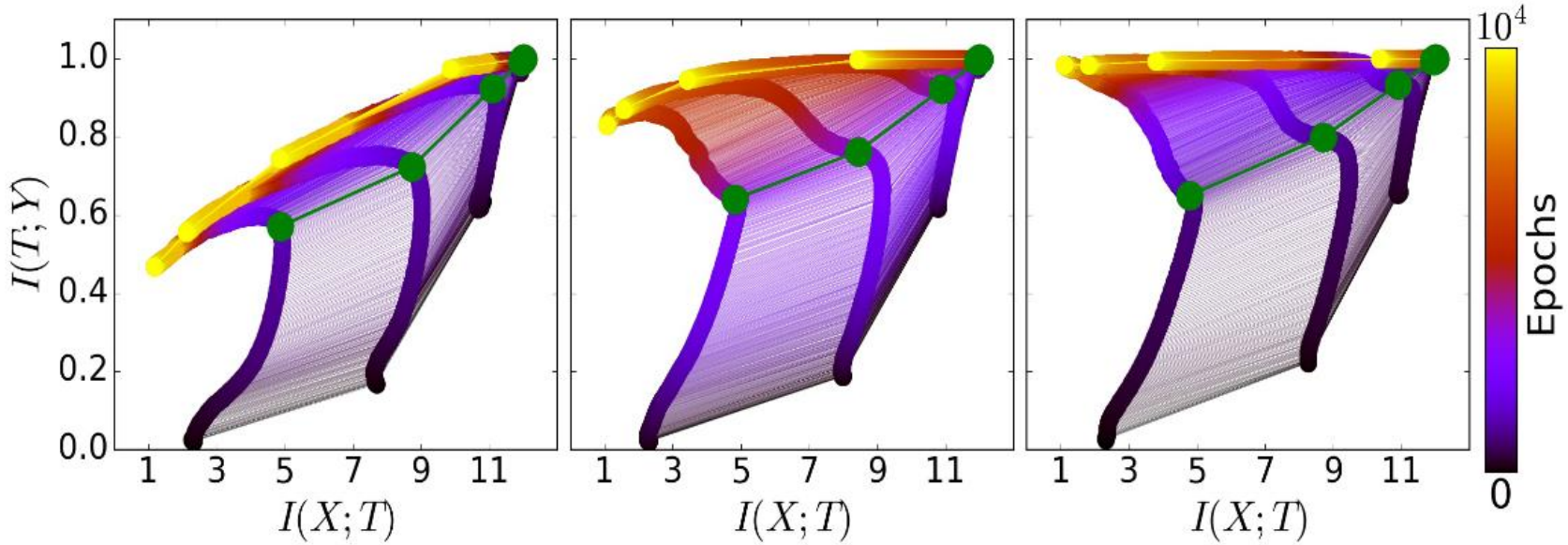


Figure 3: The evolution of the layers with the training epochs in the information plane, for different training samples. On the left - 5% of the data, middle - 45% of the data, and right - 85% of the data. The colors indicate the number of training epochs with Stochastic Gradient Descent from 0 to 10000. The network architecture was fully connected layers, with widths: input=12-10-8-6-4-2-1=output. The examples were generated by the spherical symmetric rule described in the text. The green paths correspond to the SGD drift-diffusion phase transition - grey line on Figure 4

The layers' SG distributions

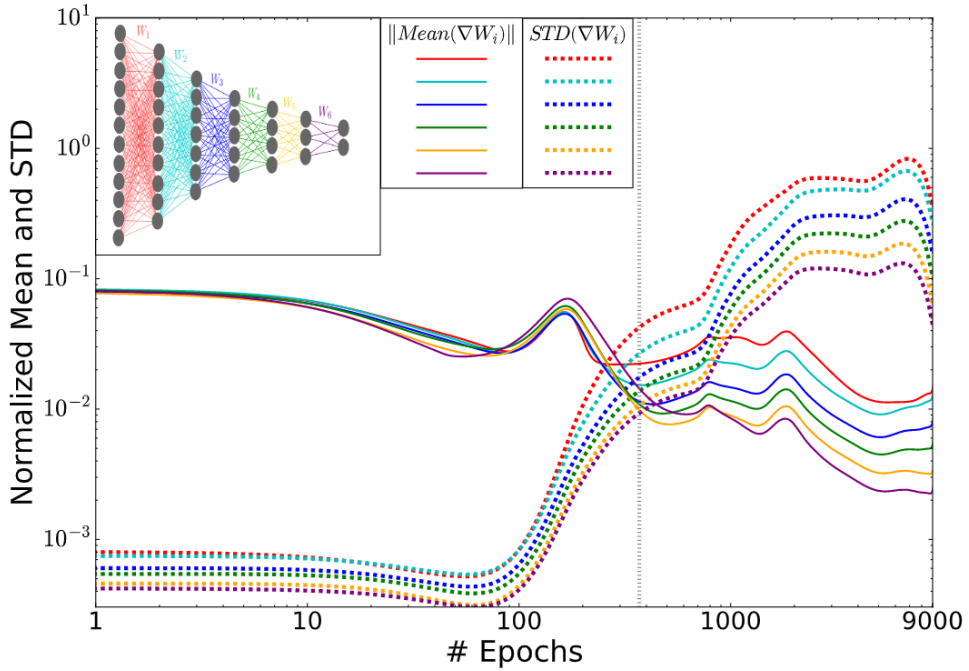


Figure 4: **The layers' Stochastic Gradients distributions during the optimization process.** The norm of the means and standard deviations of the weights gradients for each layer, as function of the number of training epochs (in log-log scale). The values are normalized by the L2 norms of the weights for each layer, which significantly increase during the optimization. The grey line ($\sim$ 350 epochs) marks the transition between the first phase, with large gradient means and small variance (*drift*, high gradient SNR), and the second phase, with large fluctuations and small means (*diffusion*, low SNR). Note that the gradients log (SNR) (the log differences between the mean and the STD lines) approach a constant for all the layers, reflecting the convergence of the network to a configuration with constant flow of relevant information through the layers!

The diffusion processes can be described by a Focker-Planck equation [see e.g. Risken (1989)], whose stationary distribution maximizes the entropy of the weights distribution, under the training error constraint.
That in turn maximizes the conditional entropy, $H(X|T_i)$, or minimizes the mutual information $I(X;T_i) = H(X) - H(X|T_i)$, because the input entropy, $H(X)$, does not change.

The computational benefit of the hidden layers

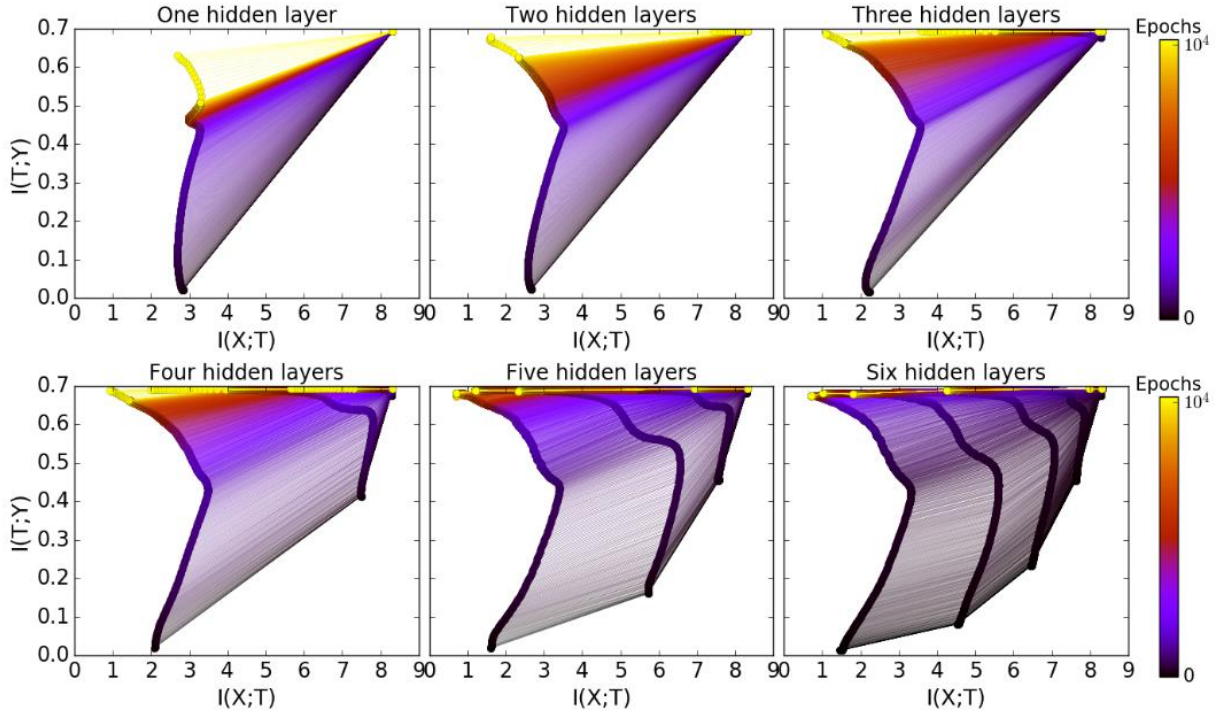


Figure 5: **The layers information paths during the SGD optimization for different architectures.** Each panel is the *information plane* for a network with a different number of hidden layers. The width of the hidden layers start with 12, and each additional layer has 2 fewer neurons. The final layer with 2 neurons is shown in all panels. The line colors correspond to the number of training epochs.

1. Adding hidden layers dramatically reduces the number of training epochs for good generalization.
2. The compression phase of each layer is shorter when it starts from a previous compressed layer.
3. The compression is faster for the deeper layers. Whereas in the drift phase the lower layers move first, in the diffusion phase the top layers compress first and ”pull” the lower layers after them. Adding more layers seems to add intermediate representations which accelerates the compression.
4. Even wide hidden layers eventually compress in the diffusion phase. Adding extra width does not help.

Evolution of the layers with training data size

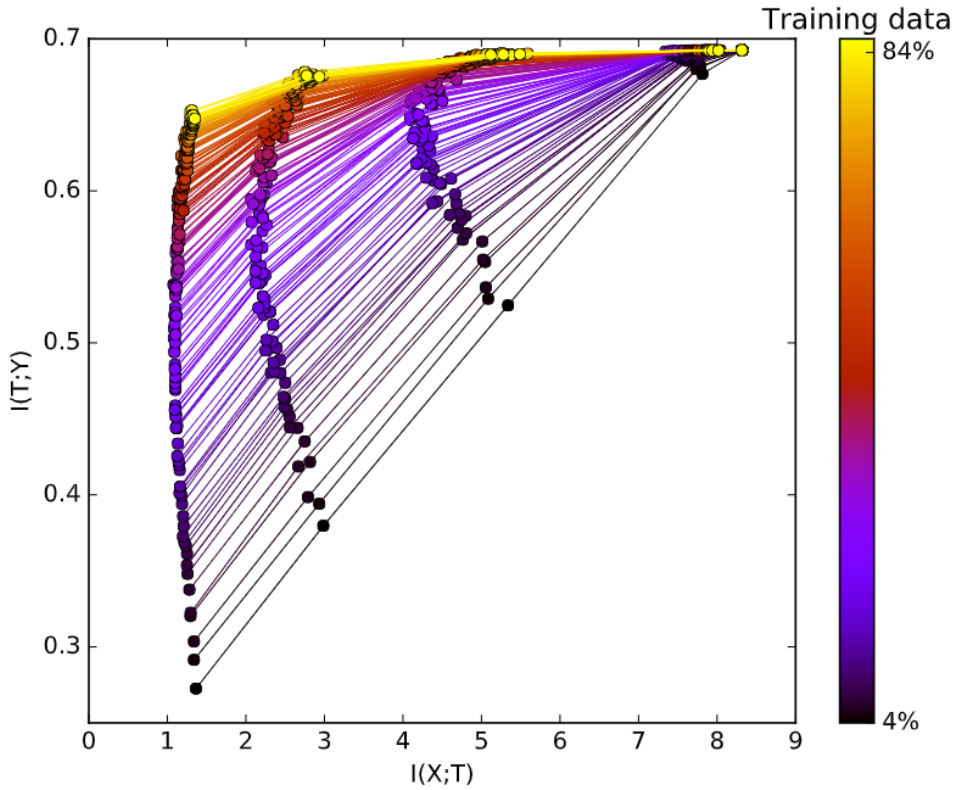


Figure 7: **The effect of the training data size on the layers in the *information plane*.** Each line (color) represents a converged network with a different training sample size. Along each line there are 6 points for the different layers, each averaged over 50 random training samples and randomized initial weights.

- With increasing training size the layers' true label information (generalization) I_Y is pushed up and gets closer to the theoretical IB bound for the rule distribution.
- For the deeper layers the network learns to preserve more of the information on Y and better compress the irrelevant information in X. With larger training samples more details on X become relevant for Y.

On The Information Bottleneck Theory Of Deep Learning

Andrew M Saxe, Yamini Bansal, Joel Dapello,

Madhu Advani, Artemy Kolchinsky, Brendan D Tracey

and David D Cox

Compression and neural nonlinearities

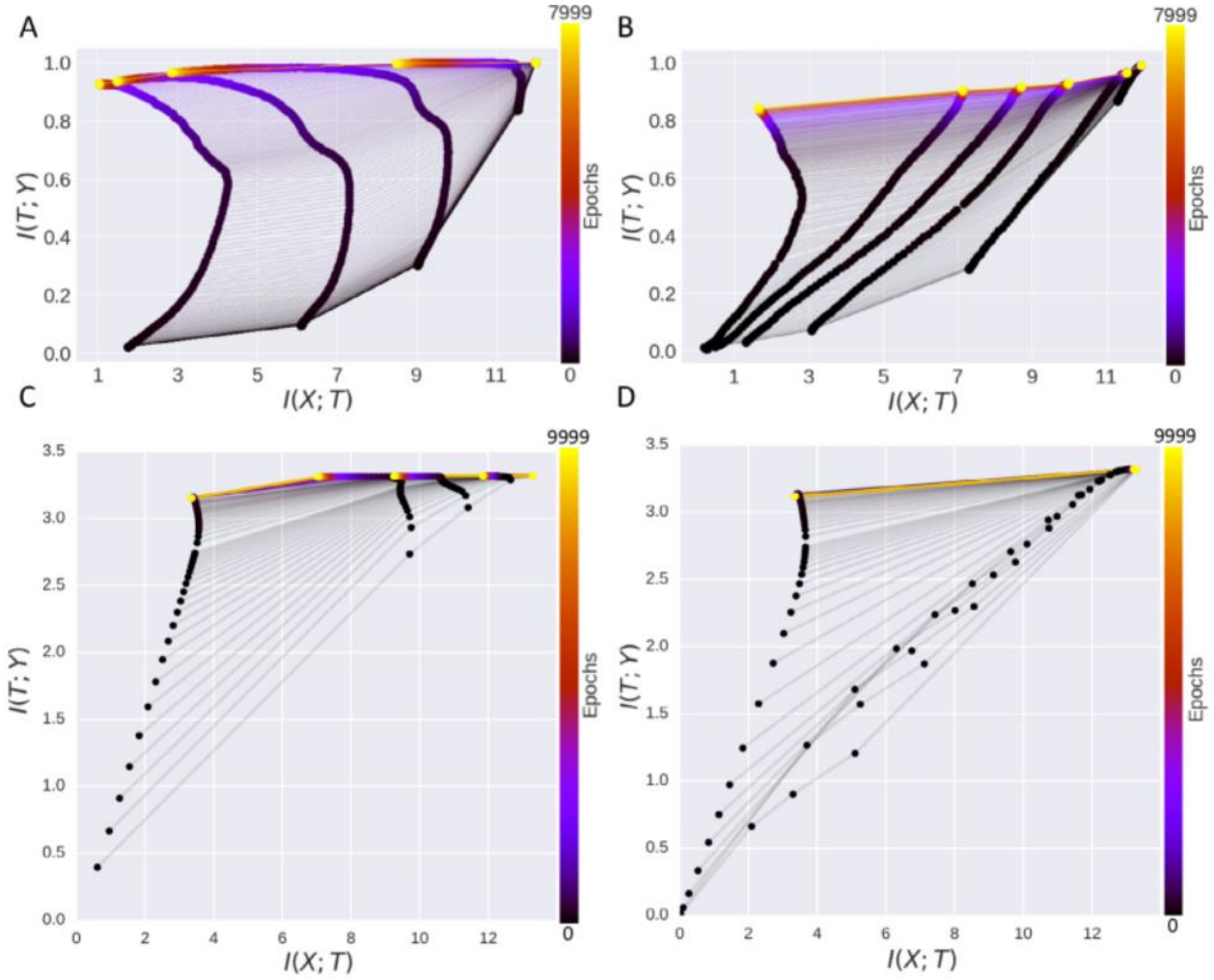


Figure 1. Information plane dynamics and neural nonlinearities. (A) Replication of Shwartz-Ziv and Tishby (2017) for a network with tanh nonlinearities (except for the final classification layer which contains two sigmoidal neurons). The x-axis plots information between each layer and the input, while the y-axis plots information between each layer and the output. The color scale indicates training time in epochs. Each of the six layers produces a curve in the information plane with the input layer at far right, output layer at the far left. Different layers at the same epoch are connected by fine lines. (B) Information plane dynamics with ReLU nonlinearities (except for the final layer of 2 sigmoidal neurons). Here no compression phase is visible in the ReLU layers. For learning curves of both networks, see appendix A. (C) Information plane dynamics for a tanh network of size 784-1024-20-20-20-10 trained on MNIST, estimated using the non-parametric kernel density mutual information estimator of Kolchinsky and Tracey (2017) and Kolchinsky *et al* (2017). (D) Information plane dynamics for a ReLU network with same configuration as panel (C). No compression is observed except in the final classification layer which contains sigmoidal neurons. See appendix B for the KDE MI method applied to the original Tishby dataset; additional results using a second popular nonparametric k-NN-based method (Kraskov *et al* 2004); and results for other neural nonlinearities.

Information plane dynamics

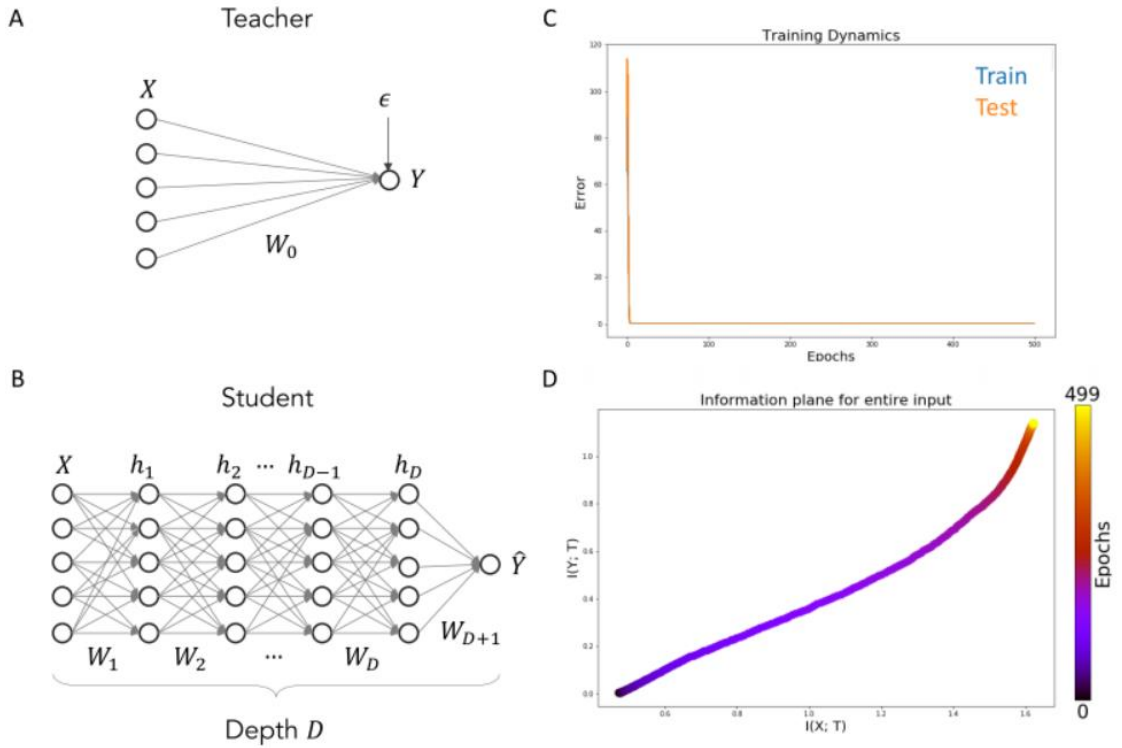


Figure 3. Generalization and information plane dynamics in deep linear networks. (A) A linear teacher network generates a dataset by passing Gaussian inputs X through its weights and adding noise. (B) A deep linear student network is trained on the dataset (here the network has 1 hidden layer to allow comparison with figure 4(A), see supplementary figure F1 for a deeper network). (C) Training and testing error over time. (D) Information plane dynamics. No compression is observed.

Simultaneous fitting and compression

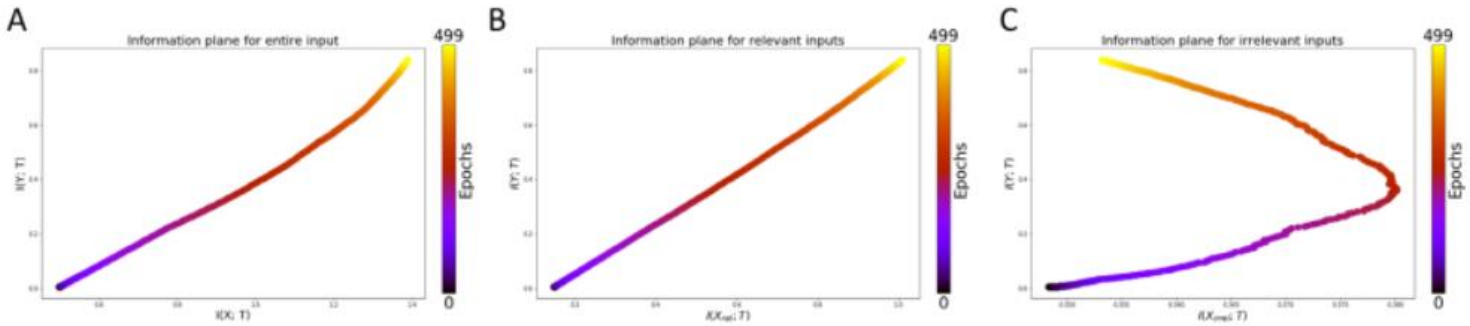


Figure 6. Simultaneous fitting and compression. (A) For a task with a large task-irrelevant subspace in the input, a linear network shows no overall compression of information about the input. (B) The information with the task-relevant subspace increases robustly over training. (C) However, the information specifically about the task-irrelevant subspace does compress after initially growing as the network is trained.

When a task requires ignoring some inputs, the information with these inputs specifically will indeed be reduced; but overall mutual information with the input in general may still increase.

Deep Variational Information Bottleneck

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, Kevin Murphy

Approach

max $I(Z,Y) - \beta I(Z,X)$

$$I(Z,Y) = \int dy\, dz\, p(y,z) \log \frac{p(y,z)}{p(y)p(z)} = \int dy\, dz\, p(y,z) \log \frac{p(y|z)}{p(y)}\,.$$

where $p(y|z)$ is fully defined by our encoder and Markov Chain as follows:

$$p(y|z) = \int dx\, p(x,y|z) = \int dx\, p(y|x)p(x|z) = \int dx\, \frac{p(y|x)p(z|x)p(x)}{p(z)}.$$

let $q(y|z)$ be a variational approximation to $p(y|z)$

$$\mathrm{KL}[p(Y|Z), q(Y|Z)] \geq 0 \implies \int dy\, p(y|z) \log p(y|z) \geq \int dy\, p(y|z) \log q(y|z)\,,$$

and hence

$$\begin{aligned} I(Z,Y) &\geq \int dy\, dz\, p(y,z) \log \frac{q(y|z)}{p(y)} \\ &= \int dy\, dz\, p(y,z) \log q(y|z) - \int dy\, p(y) \log p(y) \\ &= \int dy\, dz\, p(y,z) \log q(y|z) + H(Y)\,. \end{aligned}$$

Following standard practice in the IB literature, we assume that the joint distribution $p(X,Y,Z)$ factors as follows:

$$p(X,Y,Z) = p(Z|X,Y)p(Y|X)p(X) = p(Z|X)p(Y|X)p(X) \tag{5}$$

i.e., we assume $p(Z|X,Y) = p(Z|X)$, corresponding to the Markov chain $Y \leftrightarrow X \leftrightarrow Z$.

$$p(y,z) = \int dx\, p(x,y,z) = \int dx\, p(x)p(y|x)p(z|x)$$

$$I(Z,Y) \geq \int dx\, dy\, dz\, p(x)p(y|x)p(z|x) \log q(y|z)\,.$$

Approach

max $I(Z,Y) - \beta I(Z,X)$

$$I(Z,X) = \int dz\, dx\, p(x,z) \log \frac{p(z|x)}{p(z)} = \int dz\, dx\, p(x,z) \log p(z|x) - \int dz\, p(z) \log p(z)$$

let $r(z)$ be a variational approximation to $p(z)$

$$\mathrm{KL}[p(Z), r(Z)] \geq 0 \implies \int dz\, p(z) \log p(z) \geq \int dz\, p(z) \log r(z)$$

$$I(Z,X) \leq \int dx\, dz\, p(x)p(z|x) \log \frac{p(z|x)}{r(z)}$$

$$I(Z,Y) - \beta I(Z,X) \geq \int dx\, dy\, dz\, p(x)p(y|x)p(z|x) \log q(y|z) - \beta \int dx\, dz\, p(x)p(z|x) \log \frac{p(z|x)}{r(z)} = L\,.$$

We can approximate $p(x,y) = p(x)p(y|x)$ using the empirical data distribution $p(x,y) = \frac{1}{N}\sum_{n=1}^{N} \delta_{x_n}(x)\delta_{y_n}(y)$

$$L \approx \frac{1}{N} \sum_{n=1}^{N} \left[\int dz\, p(z|x_n) \log q(y_n|z) - \beta\, p(z|x_n) \log \frac{p(z|x_n)}{r(z)} \right].$$

Suppose we use an encoder of the form $p(z|x) = \mathcal{N}(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$, where f_e is an MLP which outputs both the K-dimensional mean μ of z as well as the $K \times K$ covariance matrix Σ. Then we can use the reparameterization trick (Kingma & Welling, 2014) to write $p(z|x)dz = p(\epsilon)d\epsilon$, where $z = f(x,\epsilon)$ is a deterministic function of x and the Gaussian random variable ϵ.

min

$$J_{IB} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[-\log q(y_n | f(x_n, \epsilon)) \right] + \beta\, \mathrm{KL}\left[p(Z|x_n), r(Z) \right]$$

$$\left[D_{\mathrm{KL}}(P_\theta(M|X) \| R(M)) - D_{\mathrm{KL}}(P_\theta(M) \| R(M)) \leq D_{\mathrm{KL}}(P_\theta(M|X) \| R(M)) \right]$$

Classification results

Model	error
Baseline	1.38%
Dropout	1.34%
Dropout (Pereyra et al., 2017)	1.40%
Confidence Penalty	1.36%
Confidence Penalty (Pereyra et al., 2017)	1.17%
Label Smoothing	1.40%
Label Smoothing (Pereyra et al., 2017)	1.23%
VIB ($\beta = 10^{-3}$)	**1.13%**

Table 1: Test set misclassification rate on permutation-invariant MNIST using $K = 256$. We compare our method (VIB) to an equivalent deterministic model using various forms of regularization. The discrepancy between our results for confidence penalty and label smoothing and the numbers reported in (Pereyra et al., 2017) are due to slightly different hyperparameters.

Results of error rate and mutual information

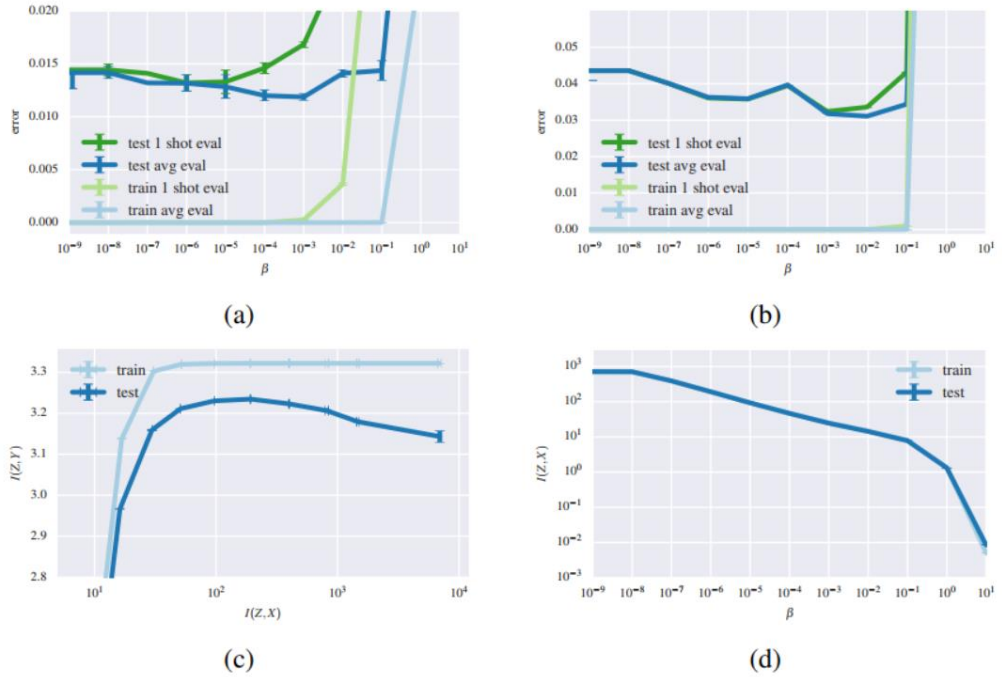


Figure 1: Results of VIB model on MNIST. (a) Error rate vs β for $K = 256$ on train and test set. "1 shot eval" means a single posterior sample of z, "avg eval" means 12 Monte Carlo samples. The spike in the error rate at $\beta \sim 10^{-2}$ corresponds to a model that is too highly regularized. Plotted values are the average over 5 independent training runs at each β. Error bars show the standard deviation in the results. (b) Same as (a), but for $K = 2$. Performance is much worse, since we pass through a very narrow bottleneck. (c) $I(Z, Y)$ vs $I(Z, X)$ as we vary β for $K = 256$. We see that increasing $I(Z, X)$ helps training set performance, but can result in overfitting. (d) $I(Z, X)$ vs β for $K = 256$. We see that for a good value of β, such as 10^{-2}, we only need to store about 10 bits of information about the input.

Results of adversarial robustness

Metric	Determ	IRv2	VIB(0.01)
Sucessful target	1.0	1.0	**0.567**
L_2	6.45	14.43	**43.27**
L_∞	0.18	0.44	**0.92**

Table 2: Quantitative results showing how the different Inception Resnet V2-based architectures (described in Section 4.2.5) respond to targeted L_2 adversarial examples. *Determ* is the deterministic architecture, *IRv2* is the unmodified Inception Resnet V2 architecture, and *VIB(0.01)* is the VIB architecture with $\beta = 0.01$. *Successful target* is the fraction of adversarial examples that caused the architecture to classify as the target class (soccer ball). Lower is better. L_2 and L_∞ are the average L distances between the original images and the adversarial examples. Larger values mean the adversary had to make a larger perturbation to change the class.

Nonlinear Information Bottleneck

Artemy Kolchinsky, Brendan D. Tracey and David H. Wolpert

Approach

$$I_\theta(X;M) = D_{\mathrm{KL}}(P_\theta(M|X)\|R(M)) - D_{\mathrm{KL}}(P_\theta(M)\|R(M)) \leq D_{\mathrm{KL}}(P_\theta(M|X)\|R(M))$$

(Variable IB)

$$I_\theta(X;M) \leq \hat{I}_\theta(X;M) := -\frac{1}{N}\sum_i \log \frac{1}{N}\sum_j e^{-D_{\mathrm{KL}}\left[\mathcal{N}(f_\theta(x_i),\Sigma_\theta(x_i))\middle\|\mathcal{N}(f_\theta(x_j),\Sigma_\theta(x_j))\right]}.$$

(Nonlinear IB)

Kolchinsky, A.; Tracey, B.D. Estimating Mixture Entropy with Pairwise Distances. Entropy 2017, 19, 361.

Results on MNIST

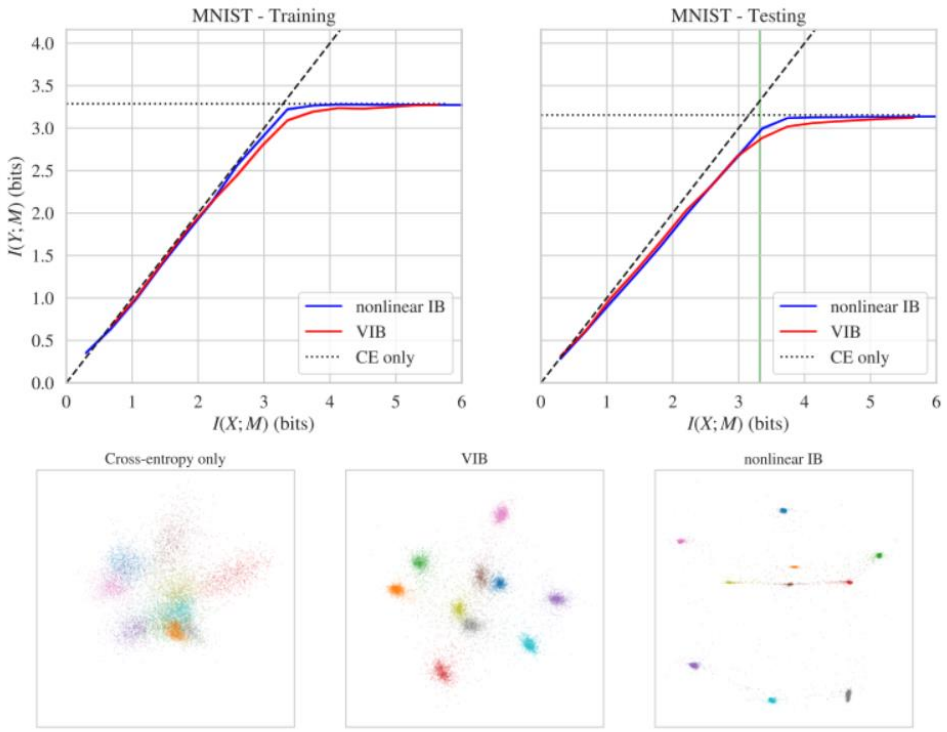


Figure 1. Top row: Info-plane diagrams for nonlinear IB and variational IB (VIB) on the MNIST training (**left**) and testing (**right**) data. The solid lines indicate means across five runs, shaded region indicates the standard error of the mean. The black dashed line is the data-processing inequality bound $I(Y;M) \leq I(X;M)$, the black dotted line indicates the value of $I(Y;M)$ achieved by a baseline model trained only to optimize cross-entropy. **Bottom row**: Principal component analysis (PCA) projection of bottleneck layer activity (on testing data, no noise) for models trained with regular cross-entropy loss (**left**), VIB (**middle**), and nonlinear IB (**right**) objectives. The location of the nonlinear IB and VIB models shown in the bottom row is indicated with the green vertical line in the top right panel.

Results on FashionMNIST

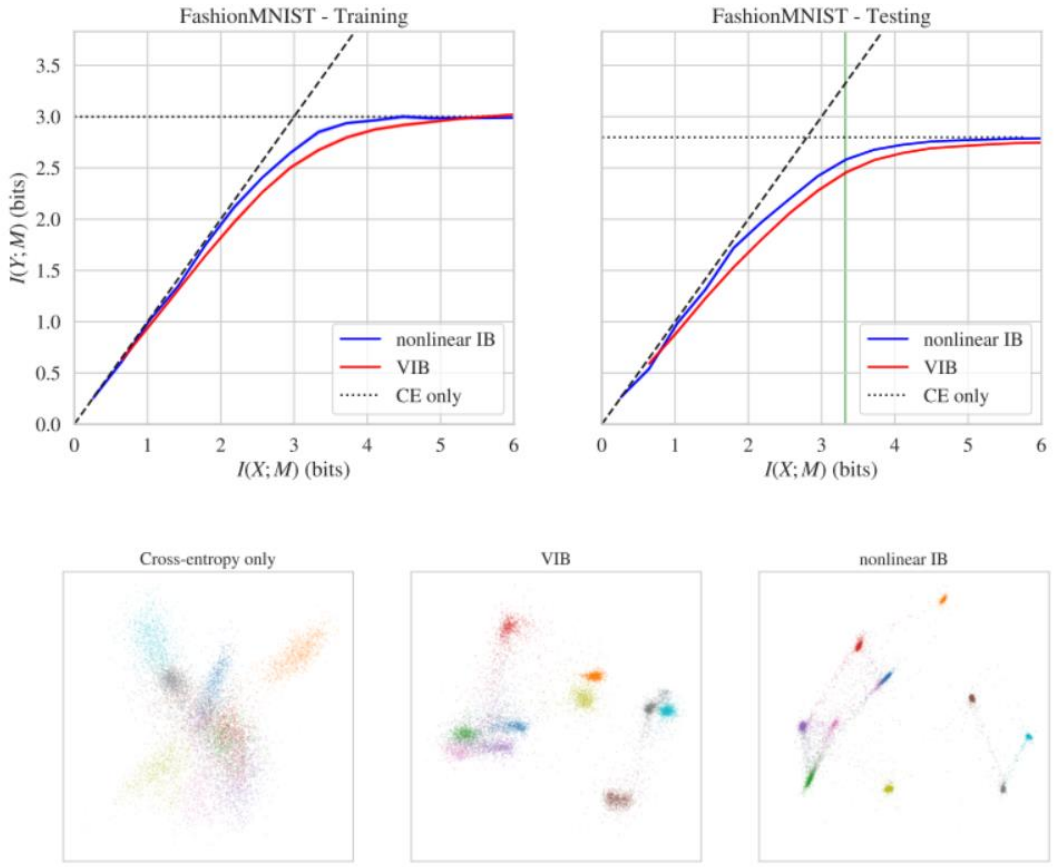


Figure 2. Top row: Info-plane diagrams for nonlinear IB and VIB on the FashionMNIST dataset. **Bottom row**: PCA projection of bottleneck layer activations for models trained only to optimize cross-entropy (**left**), VIB (**middle**), and nonlinear IB (**right**) objectives. See caption of Figure 1 for details.

Results on FashionMNIST

Table 1. Amount of prediction $I(Y;M)$ achieved at compression level $I(X;M) = \log 10$ for both nonlinear IB and VIB.

Dataset		Nonlinear IB	VIB
MNIST	Training	**3.22**	3.09
	Testing	**2.99**	2.88
FashionMNIST	Training	**2.85**	2.67
	Testing	**2.58**	2.46
California housing	Training	**1.37**	1.26
	Testing	**1.13**	1.07

Conclusions

- The information bottleneck may bring us better performance with a simple structure.
- The information bottleneck is much useful for speech information factorization.

- The most important part of learning is actually forgetting.

—— Naftali Tishby